Machine Learning Assisted Low-Thrust Orbit-Raising: A Comparative Assessment of a Sequential Algorithm and a Deep Reinforcement Learning Approach

Atri Dutta*, Adrian Arustei†, Matthew Chace‡, Pardhasai Chadalavada§, and James Steck¶,
Wichita State University, Wichita, KS, 67260.

S.M. Talha Zaidi‖ and Arslan Munir**
Kansas State University, Manhattan, KS, 66506.

The focus of the paper is machine-learning-assisted computation of low-thrust orbit-raising trajectories. We consider a sequential algorithm for computing multi-revolution trajectories, whose optimization cost function parameters can be updated through a high-level planner utilizing a suitably trained artificial neural network. Considering two different orbit-raising mission scenarios based on the final target orbit (geostationary and near-rectilinear halo orbit), we conduct numerical simulations to compare the results of this approach with that provided by a deep reinforcement learning framework.

I. Introduction

Advances in solar-electric propulsion technology in recent decades have led to its wider incorporation in a variety of space missions: interplanetary missions, geocentric missions, as well as cislunar missions. Owing to the significant mass savings enabled by such propulsion systems, their adoption by the space industry is only expected to grow in the future. One of the challenges introduced by solar-electric proplusion systems is the complex mission planning process, because the low-thrust nature of the underlying maneuvers requires the solution of a non-convex, non-linear optimal control problem. In the presence of eclipses, such as during geocentric orbit-raising, accounting for the resulting intermittent loss of power within the optimization process adds to the challenges. Furthermore, for cislunar missions, the richer and highly nonlinear dynamics owing to the presence of two celestial bodies add to the complexity of the underlying optimal control problem. The focus of this paper is the multi-revolution low-thrust orbit-raising problem that needs the above-mentioned challenges to be addressed. Specifically, we will consider two types of missions, each starting from a super-synchronous geosynchronous transfer orbit (GTO). In the first case, we consider the terminal orbit to be geocentric equatorial orbit (GEO). In the second case, we consider the terminal orbit to be a near-rectilinear halo orbit. The main distinction between the two cases is that, owing to the traversal of cislunar space by the spacecraft, the second case necessitates the consideration of the presence of gravitational fields of both the Earth and the Moon in the spacecraft dynamic model.

Computation of an optimal low-thrust orbit-raising trajectory has been a widely studied problem over the past decades and a variety of techniques have been developed; a majority of these techniques can be classified as either direct or indirect. Techniques based on indirect optimization start with writing out the necessary conditions of optimality using the calculus of variations, leading to the formulation of a two-point boundary value problem that is solved numerically to yield the optimal solution [1–4]. Alternatively, direct techniques rely on direct transcription and collocation to convert the optimal control problem to a parameter optimization problem that can be readily solved using efficient commercially available nonlinear programming solvers [5–8]. Although both techniques inherently rely on user-provided initial guess solutions for unknown variables in the problem to kick-start the numerical solution process of the underlying optimization scheme, it is generally accepted that direct methods demonstrate better numerical convergence in contrast to their indirect counterparts [9]. Nevertheless, in search of better guarantees on the convergence of numerical algorithms

*Associate Professor, Aerospace Engineering, 1845 Fairmount St Box 42, and AIAA Senior Member.
†PhD Student, Aerospace Engineering, 1845 Fairmount St Box 42, and AIAA Member.
‡MS Student, Aerospace Engineering, 1845 Fairmount St Box 42, and AIAA Member.
§Former PhD Student, Aerospace Engineering, 1845 Fairmount St Box 42, and AIAA Student Member
¶Professor, Aerospace Engineering, 1845 Fairmount St Box 42, and AIAA Associate Fellow.
‖PhD Student, Computer Science, 1119 Engineering Hall, 1701D Platt St.
**Associate Professor, Computer Science, 2162 Engineering Hall, 1701D Platt St.

associated with low-thrust orbit-raising trajectory optimization, several alternative approaches have been researched as well, such as shape-based techniques [10–13], Q-law [14, 15], or semi-analytical approaches [16–23]. Dynamic models used in the solution of the optimal control problem also influence the convergence characteristics of the numerical algorithms, and a detailed study is provided in Ref. [24]; it is generally understood that regularized dynamic models consisting of mostly "slow" variables are better suited for the numerical convergence. Slow variables are typically constant for Keplerian motion and only vary slowly in the presence of small perturbations or low thrust. Recent advances in machine learning have also seen the application of such techniques to low-thrust optimization problem from the perspective of trajectory design as well as onboard guidance [25, 26]. Autonomous guidance for multi-revolution low-thrust orbit-transfer using reinforcement learning was investigated in Ref [27]. A cascaded reinforcement learning technique was developed in Ref. [28] for determining orbit-raising maneuvers to GEO, starting from GTO or super-GTO. Reinforcement learning approaches have also been investigated for computing low-thrust trajectories in multi-body regimes, including cislunar scenarios [29, 30]. Reinforced Lyapunov control based design of low-thrust transfers have also been studied [31, 32].

Recently, a new approach to solving the low-thrust orbit-raising problem has been considered, by first creating a sequence of simpler optimization sub-problems. We refer to this approach as the sequential algorithm. The formulation of a sequential low-thrust orbit-raising problem, along with the use of a regularized dynamic model using the (h, e) orbital element set to describe the spacecraft translation dynamics, allow for an orbit-raising trajectory computation in a fast and robust manner, without relying on any user-provided initial guess [33]. This methodology enables the automated computation of low-thrust orbit-raising trajectories in the presence of eclipses; however, the above-mentioned benefits of the approach are obtained by trading off the optimality of the computed solution. One of the key contributors to the sub-optimality of the computed solutions is the fact that the optimal control subproblem in the sequential approach only considers a short-sized objective function of minimizing the deviation from the target orbit, rather than capturing any elements of a global objective function such as transfer time or fuel expenditure [34]. The short-sighted objective takes into account three different components that measure the deviation of different orbital elements weighted using an ad hoc choice of weights set by the mission designer. The weights of the objective function affect the solution of the problem [35], and can be tuned or modified in order to improve a desired performance metric [34].

In this paper, we present new results related to the sequential algorithm and also new results from the cascaded deep reinforcement learning methodology. First, we present an overview of the sequential algorithm. This algorithm has typically been used to determine transfers to GEO [33, 36]. Here, we present a numerical simulation for transfer from a super-GTO to near-rectilinear halo orbit (NRHO). Second, we show how an artificial neural network (ANN) based transfer time prediction can be used to drive a weight adaptation strategy to improve the solution of the sequential algorithm. This benefit is demonstrated for representative cases of transfers to GEO, strarting from sub-GTO, GTO, and super-GTO. We also present initial results from our current work on creating better quality artificial neural network that can lead to future improvements of the weight adaptation strategy. Finally, we demonstrate the application of a cascaded deep reinforcement learning framework for a cislunar mission. This technique was previously used to compute near-optimal transfers to GEO [28]; in this paper, we present results for a transfer from super-GTO to NRHO. Finally, we provide a comparison of the sequential and reinforcement learning framework.

II. Sequential Algorithm

In this section, we provide an overview of the sequential algorithm that solves a sequence of optimal control sub-problems, in lieu of a single optimal control problem. The algorithm also leverages the use of a regularized dynamic model that simplifies the optimization process. We first provide an overview of the dynamic model, then provide a model of the orbit-raising maneuver leading to the sequential algorithm, and also illustrate the methodology with an example.

A. Spacecraft Dynamics

In order to describe the translational dynamics of the spacecraft with solar-electric propulsion onboard, let us denote by $\boldsymbol{x} \in \mathbb{R}^6$ the states of the spacecraft, consisting of the following: specific angular momentum h, components of the specific angular momentum vector (h_X and h_Y) along the X and Y axes of the geocentric inertial reference frame $\mathcal{I}$, the components of the eccentricity vector (e_x and e_y) along the x and y axes in an orbital reference frame $\mathcal{O}$ obtained after a 2-1 rotation applied to inertial frame $\mathcal{I}$, and the angle ϕ indicating the position of the spacecraft in its orbit, with the angle being measured from the line of intersection of the orbital plane and the X-Z plane of the frame $\mathcal{I}$. While formulating the equations of motion, we consider the usual problem setting of the two-body problem describing the motion of the spacecraft in the gravitational field of the Earth; however, we allow for the inclusion of

other force/acceleration terms arising out of the spacecraft thrust as well as other celestial bodies. The equations of motion are then written as [33]:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(x) + \mathbb{G}(x)\left(\boldsymbol{u} + \boldsymbol{p}_{\text{other}}\right), \tag{1}$$

where $\boldsymbol{u} \in \mathbb{R}^3$ is the control vector denoting the magnitude and direction of the thrust force from the solar-electric propulsion, $\boldsymbol{f} \in \mathbb{R}^6$ is the Keplerian rate of change of state variables (that is, capturing only the effect of the primary gravitational body), $\boldsymbol{p}_{\text{other}} \in \mathbb{R}^3$ is a vector denoting all other forces (such as J_2 perturbation and third body gravitation) acting on the spacecraft, and $\mathbb{G} \in \mathbb{R}^{6\times3}$ is a mapping matrix that captures the effect of all non-Keplerian forces (thrust and others) to the rate of the change of the state variables $\boldsymbol{x} \in \mathbb{R}^6$. The following therefore completes the description of the dynamics of the spacecraft [33]:

$$\boldsymbol{x} = \begin{bmatrix} h \\ h_X \\ h_Y \\ e_x \\ e_y \\ \phi \end{bmatrix}, \quad \boldsymbol{f} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{\mu^2 B^2}{h^3} \end{bmatrix}, \quad A = e_x \sin\phi - e_y \cos\phi, \quad B = 1 + e_x \cos\phi + e_y \sin\phi \tag{2}$$

$$\mathbb{G} = \begin{bmatrix} 0 & \frac{h^2}{\mu m B} & 0 \\ 0 & \frac{h h_x}{\mu m B} & \frac{h^2\sqrt{h^2-h_x^2-h_y^2}}{\mu m B\sqrt{h^2-h_y^2}}\sin\phi + \frac{h h_x h_y}{\mu m B\sqrt{h^2-h_y^2}}\cos\phi \\ 0 & \frac{h h_y}{\mu m B} & -\frac{h\sqrt{h^2-h_y^2}}{\mu m B}\cos\phi \\ \frac{h\sin\phi}{\mu B} & \frac{2h\cos\phi}{\mu m} + \frac{hA\sin\phi}{\mu m B} & \frac{h e_y h_y \sin\phi}{\mu m B\sqrt{h^2-h_y^2}} \\ -\frac{h\cos\phi}{\mu B} & \frac{2h\sin\phi}{\mu m} - \frac{hA\cos\phi}{\mu m B} & -\frac{h e_x h_y \sin\phi}{\mu m B\sqrt{h^2-h_y^2}} \\ 0 & 0 & -\frac{h h_y \sin\phi}{\mu m B\sqrt{h^2-h_y^2}} \end{bmatrix}. \tag{3}$$

A couple of remarks with respect to the dynamic model need to be noted. First, the above-mentioned equations are written by considering time as the independent variable. For convenience of the optimization, the equations are often written by posing the only fast variable ϕ as the independent variable. Second, for solving minimum-propellant problems for cislunar missions, the time and mass variables are also included within the state vector, in order to allow for consideration of coasting and ephemeris models respectively.

B. Optimal Control Sub-problem

We have summarized the idea behind the sequential algorithm in Figure 1. As shown in the figure, the spacecraft moves from an initial state 'O' (blue) to a final state 'F' (green). Instead of solving a complex optimal control problem (OCP) that generates a path from O to F (represented by the blue arrow), the optimization problem is broken into a number of steps. Over each step, the spacecraft transitions through a set of intermediate states (I_1, I_2, I_3, I_4, and I_5 marked in red in the figure); these intermediate states are not known *a priori*, but are determined by the solution of optimization sub-problems (marked S_1, S_2, S_3, S_4, S_5, and S_6 in the figure), each of which generates a path only up to the next state. This next state is labeled as an intermediate state, unless some pre-defined terminal conditions are met, indicating the attainment of the final state F. Three important remarks need to be mentioned here for clarity of the sequential methodology considered in this paper:

- The number of steps to reach the final state 'F' is not known *a priori* and is determined once the subproblems are solved till the terminal conditions are met.
- To facilitate the computation of the trajectory by solving a sub-problem, we define a sub-problem planning period in an *ad hoc* manner, usually considered to be one revolution (a change of the angle ϕ by an amount 2π.
- The sequential algorithm is different from a traditional receding horizon scheme in the sense that the path computed during each planning period is an incomplete trajectory not leading to F (except for the final sub-problem), as opposed to computation of the entire trajectory during each planning period.

As already mentioned, we define each planning horizon over one revolution of the spacecraft. Let us use the index k to denote a planning horizon, where $1 \leq k \leq N$ and N being the number of planning horizons needed for the transfer to

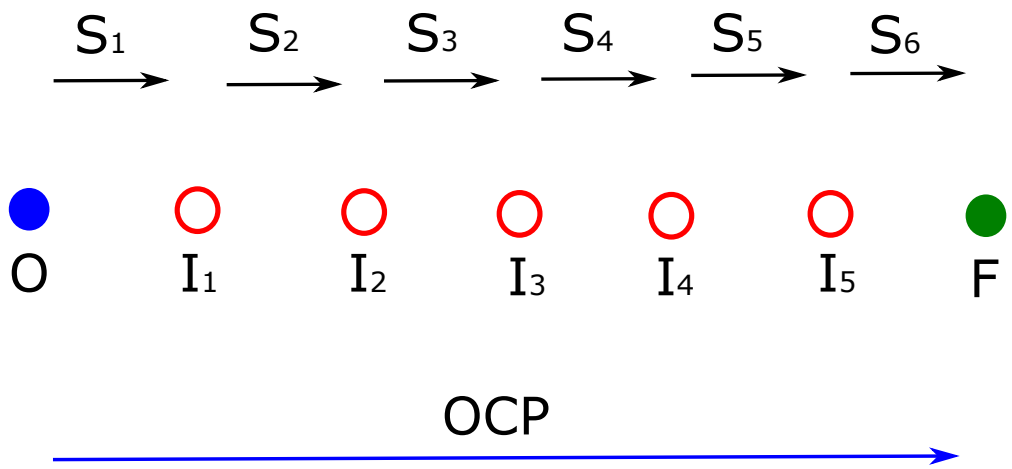

Fig. 1 Schematic overview of sequential algorithm.

complete. Recall that, N is not known *a priori*. For the k-th planning horizon, let us denote the states at the beginning of the revolution by $\boldsymbol{x}_{k-1}$ and at the end of the revolution by $\boldsymbol{x}_k$. The transition of the states from $\boldsymbol{x}_{k-1}$ to $\boldsymbol{x}_k$ is determined by the solution of the optimal control sub-problem that aims to to minimize the deviation of the spacecraft from the pre-specified target orbit (which can be the GEO or NRHO depending on the relevant mission application). To measure this deviation, we denote the state of the spacecraft corresponding to the target orbit by $\boldsymbol{x}_{\text{target}}$. The deviation from the target state at the end of the revolution is given by:

$$J_k = \left(\boldsymbol{x}_k - \boldsymbol{x}_{\text{target}}\right)^T \mathbb{W} \left(\boldsymbol{x}_k - \boldsymbol{x}_{\text{target}}\right), \tag{4}$$

We therefore write the optimal control sub-problem as the minimization of the above objective function, subject to Equations (1)–(3), and the initial states being given by $\boldsymbol{x}_k$. The matrix $\mathbb{W}$ is a diagonal matrix consisting of weights for different components of the objective function. The weights are often selected in an *ad hoc* manner, enforcing the convexity constraint that the sum of the diagonal elements, given by $diag(\mathbb{W})$, is equal to 1. The sequence of optimal control sub-problems are solved until a set of terminal conditions, denoted by $\mathcal{S}$, are satisfied. Specifically, $\boldsymbol{x} \in \mathcal{S} \subset R^6$ *if and only if* the following are met:

$$0 \leq \left(\boldsymbol{x}_k - \boldsymbol{x}_{\text{target}}\right)^T \mathbb{W} \left(\boldsymbol{x}_k - \boldsymbol{x}_{\text{target}}\right) \leq \zeta_{\text{tol}}, \tag{5}$$

where ζ_{tol} is a pre-defined tolerance signifying how closely the attainment of the target orbit needs to be satisfied.

C. Optimal Control Sub-Problem

Within the framework of the sequential algorithm, a sequence of optimal control sub-problems are solved in order to determine the thrust direction angles such that the objective function given by Eq. (4) is minimized. The thrust magnitude is set as constant to the maximum available amount, noting that there is no thrust available when the spacecraft passes through the shadow of the Earth. The control variables are parameterized using B-splines, therefore transforming the continuous decision variables into a discrete set. A 'Matlab' noninear programming solver `fmincon`, which solves constrained optimization problems, is utilized in order to solve this optimization sub-problem. The equations of motion are numerically integrated, and initial values of the thrust direction angles for the first revolution are set to zero. For future revolutions, the thrust angles are set to those obtained from the optimization associated with the previous revolution. Because of the nature of the objective function being short-sighted in nature (not capturing a global objective such as transfer time or minimum fuel), the solution yielded by the sequential algorithm is sub-optimal in nature. However, one can use the obtained trajectory from the sequential algorithm to yield a good initial guess to construct a good quality local minimum solution for both minimum time and minimum propellant solution. An important remark that needs to be mentioned here is that for traditional orbit-raising problems to GEO, the sequential solution propagates in a forward manner. However, for studying cislunar transfer problems with rendezvous-like constraints, it is beneficial to apply the sequential algorithm in a reverse manner that involves back-stepping in time. In this case, the sequential methodology is continued until the initial conditions of the transfer are met.

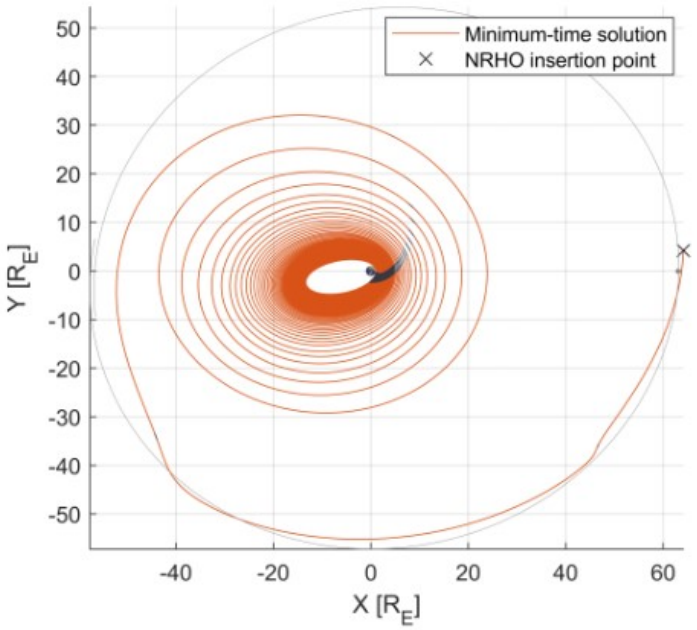

Fig. 2 Minimum time trajectory from super-GTO to NRHO (projection on X-Y plane).

Example 1. Orbit-raising from super-GTO to NRHO.

We consider a spacecraft with the following solar-electric propulsion system: thrust = 1.087N and specific impulse = 1976.8 s. The starting orbit is considered to be a supersynchronous GTO with perigee altitude of 200 km and apogee altitude of 80,000 km, inclination of 27 degrees, right ascension of ascending node of 0 degrees, argument of periapsis of 0 degrees, and true anomaly of 0 degrees. The initial epoch and mass of the spacecraft at the beginning of the transfer is set to be free. The target orbit is a 9:2 Southern L2 NRHO with perilune altitude of 3269 km, period of 6.573 day. The final targeted mass at NRHO is 7000 kg at the time of deployment, with the target epoch considered to be January 2 of 2020. This problem is actually solved in a backward manner, with the sequential algorithm applied as back-stepping in time. As already mentioned, this is in contrast to the typical 'forward' propagation methodology for computing trajectories for orbit-raising to GEO. The solution of the sequential algorithm is utilized to obtain a local minimum (transfer time) trajectory. The projection of the 178.4-day trajectory on the inertial X-Y plane is demonstrated in Figure 2.

III. Machine-Learning-Assisted Sequential Algorithm

As already mentioned, the sequential algorithm leads to sub-optimal solution, owing to the use of a short-sighted objective function for the optimal control sub-problems, in contrast to a global objective (such as transfer time or propellant expenditure). The quality of the solution generated by the sequential algorithm is affected by two important factors, namely:

- Length of the planning horizon: In a previous work, we have shown that the length of the planning horizon for the optimal control sub-problem can be modified in order to improve the result of the sequential algorithm [35],
- Weights of the objective function: In a previous work, we have shown that the weights of the objective function can be modified in order to improve the solution generated by the sequential algorithm [34].

Typically, these parameters are selected in an *ad hoc* manner, with their values set manually based on the experience of the user with the algorithm performance for various settings. Even under these ad hoc settings, the sequential algorithm can be used to generate a good quality minimum time transfers [37]. However, the attainment of a global minimum solution is not guaranteed. To this end, this paper investigates ways in which we can incorporate machine learning to select the objective function weights and improve the performance of the sequential algorithm. To this end, this paper presents some new improvements on the ANN-driven adaptive weight selection strategy, along with our current work focusing on the improvement of the performance of the ANN in predicting both the transfer time and the number of

revolutions of the transfer.

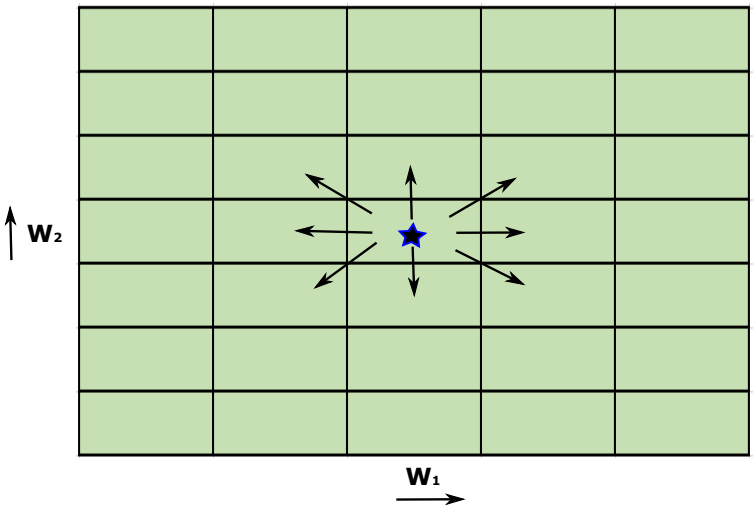

Fig. 3 Adaptive weight selection policy for improving sequential solution.

A. Adaptive Weight Strategy

The fundamental idea behind the adaptive weight strategy is that, for a given optimal control sub-problem, the vector of weights $diag\,(\mathbb{W})$ in the objective function can be changed to a neighboring weight, with a preset stepsize ϵ for each of the elements of the weight vector defining the neighborhood of the weight. Considering that the global objective is the time for the transfer, the optimal weight for the current planning horizon (revolution) is then selected to be the one in the neighborhood that will likely yield the least transfer time to reach the target. Owing to the lack of *a priori* knowledge about the transfer time for different weight selection in the neighborhood of the current weight vector, the transfer time is estimated by training an artificial neural network; this network is then incorporated within the sequential algorithm to select better weights for an optimal control sub-problem. In a previous study [34], we had compared five different weight selections from the neighborhood (current weight vector and four in the neighborhood) to determine the optimal weights for the strategy for low-thrust orbit-raising to the GEO. In this current study, we consider an extended version of the adaptive strategy by allowing for nine different weight selections within the neighborhood of the current weight vector; note that the nine includes the current weight vector. The selected weight vector for an optimal control sub-problem is the one which will yields a smaller transfer time. Figure 3 demonstrates the idea of for a three-dimensional weight vector constrained by a convexity constraint, thereby needing only two (W_1 and W_2 in the figure) of the three to be selected; the selected nine points include the current weights (indicated by the star) and the weights corresponding to the 8 neighboring cells, as illustrated in the figure. The transfer time corresponding to a given weight vector assumes that the weight vector once selected remains constant for the rest of the transfer, and the transfer time is estimated by using a neural network trained on data-points generated by running numerous instances of the sequential algorithm. The dataset generation takes advantage of the automated computation of the sequential algorithm. For instance, for an orbit-raising maneuver from super-GTO to GEO, we have used 402,623 data-points to train an ANN, yielding a mean square error of 0.124. These data-points include those obtained from both planar and non-planar transfer simulations. In the previous study [34], planar and non-planar transfer data-points were used separately to train different neural networks; the adaptive strategy would switch from using non-planar to planar ANN to estimate transfer times, resulting in jumps in the weight vector as a non-planar transfer becomes planar towards the end of the transfer to GEO. This jump in the weight vector is avoided currently through the use of a single ANN. Nevertheless, note that a single ANN for all types of orbit-raising maneuver does not typically lead to good accuracy of the predicted transfer times; hence, we use different ANNs depending on where the transfer starts, sub-GTO, GTO, and super-GTO. Herein, we discuss the results for two cases of orbit-raising from super-GTO, one planar and another non-planar. The weight vector is considered to be:

$$diag\,(\mathbb{W}) = [W_1 \quad W_3 \quad W_3 \quad W_2 \quad W_2 \quad 0]\,, \qquad \text{where } W_1 + W_2 + W_3 = 1, \tag{6}$$

with the final element of the above vector is zero because the orbit-raising problem is typically a five-state targetting problem. For the following examples, we consider a starting orbit of semi-major axis 29, 003 km and eccentricity of 0.7715, with the spacecraft having a starting mass of 5, 000 kg and a propulsion system providing a thrust force of 1.16 N and $I_{\text{sp}} = 1786$ s. A couple of remarks need to be outlined here:

- For the implementation of the sequential algorithm in general, the convexity constraint is not necessary. However, for the weight adaptation strategy, the convexity constraint reduces the number of weight variables over which the search needs to be performed.
- The weight matrix can be utilized the scale the different elements of the state vector differently, for example, one can use:

$$diag\,(\mathbb{W}) = \begin{bmatrix} W_1/h_d^2 & W_3/h_d^2 & W_3/h_d^2 & W_2 & W_2 & 0 \end{bmatrix},$$

  in order to scale the angular momentum (h, h_X, h_Y) targets separately from eccentricity targets (e_x, e_y).

Example 2. Planar transfer from super-GTO to GEO.

For Super-GTO planar transfer, initial values of W_1 and W_2 are considered to be 0.75 and 0.25 respectively. Because this is a planar case, W_3 can be zero, thereby reducing the adaptive weight strategy to be a selection of weights on an one-dimensional grid (in contrast to the two-dimensional grid in Figure 3). Hence, for the planar case, there are only three possible choices of W_1 possible at each step (noting that $W_2 = 1 - W_1$). For each possible weight combination, the transfer time is estimated using the neural network, and the weight combination that gives the least transfer time is chosen for the next sub-problem. We consider the step-size in the weight strategy ϵ to be 0.01. Trajectory generated using the adaptive mechanism with the selected step-size gives the transfer time of 102.99 sidereal days that is less compared to the transfer time of 104.02 sidereal days for the trajectory generated using the same initial weights for the entire transfer.

Example 3. Non-planar transfer from super-GTO to GEO .

For the non-planar transfer, we consider that the super-GTO now has an angle of inclination equal to 28.5 deg. In this case, the adaptive strategy focuses on choosing the combination of weights (W_1 and W_2) before solving each optimal control sub-problem. In other words, all nine possible weight combinations (as shown in Figure 3) are used to determine the optimal weights at each step. Trajectory generated using adaptive mechanism with initial value of $W_1 = 0.5$ and $W_3 = 0.4$ and with constant step-size $\epsilon = 0.01$ for all weights; the resulting trajectory yields a transfer time of 128.91 sidereal days. This transfer time is less compared to the transfer time of 137.78 sidereal days obtained for the trajectory generated using the initial weights kept constant during the entire transfer.

Example 4. Orbit-raising to GEO starting from representative GTO and sub-GTO, and comparison with super-GTO.

We consider transfers starting from representative GTO/GTO-type orbits, and have employed the adaptive weight selection strategy to compute the low-thrust transfer. In each case, reduction of transfer times was observed. Table 1 summarizes the results, and compares the transfer times obtained from the sequential algorithm (constant weights) and the adaptive weight selection strategy. It is seen that substantial reduction of transfer times can be obtained for the sequential algorithm by allowing for adaptive modification of weights of the objective function.

B. Prediction of Transfer Time and Revolutions

Although the adaptive weight strategy improves the solution computed by the sequential algorithm, the ANN utilized is trained from a dataset generated by running the sequential algorithm, leveraging the fast, robust and automated computations. However, the algorithm itself is known to compute sub-optimal trajectories; hence, there is a potential for greater improvement if the ANN can be trained on globally optimal solutions, or at least good-quality locally optimal solutions of the problem. Furthermore, we also plan to gain insight into the weight selection strategy for cislunar transfers as well, such as orbit-raising to NRHO. To this end, our current research efforts are focused on training a neural network on datasets that are good-quality locally optimal solutions of the problem. The solver used to generate this datapoints is described in Ref. [37], which uses adjoint sensitivites to improve the sequential solution and obtain a locally optimal solution. Furthermore, in the current efforts, we are also looking at the possibility of predicting the number of revolutions. Hence, the neural network has two outputs (transfer time and number of revolutions) to predict. The input vector of the neural network is also a larger set of variables that includes the state of the spacecraft, orbital elements, and weight vector (without constrained by the convexity constraint). An architecture with two hidden layers is

Table 1 Summary of Transfer Time Reduction of Sequential Algorithm for Orbit-Raising to GEO.

Initial Orbit	Propulsion system	Time (sequential) (days)	Time (adaptive) (days)	Reduction in transfer time
a=24396 km e= 0.7283 i= 28.5 deg	m_0=5000 kg F=1.16 N I_{sp}= 1786 sec	167.26	138.12	17.42 %
a= 21503km e=0.6918 i= 28.5 deg	m_0=5000 kg F=1.16 N I_{sp}= 1786 sec	160.07	146.89	8.23 %
a= 29003 km e= 0.7715 i= 28.5 deg	m_0=5000 kg F=1.16 N I_{sp}= 1786 sec	137.78	128.91	6.43%

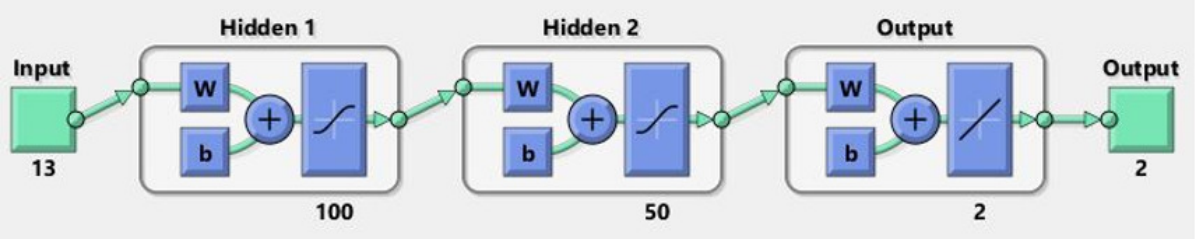

Fig. 4 Neural network architecture for predicting transfer time and number of revolutions.

used for the prediction. This is summarized in Figure 4. We have utilized a dataset in which the problem of orbit-raising to NRHO is solved using the sequential algorithm for a large selection of weights. Figure 5 depicts the variation of the mean square error during the ANN training process on this dataset of points. Our future work will focus on the

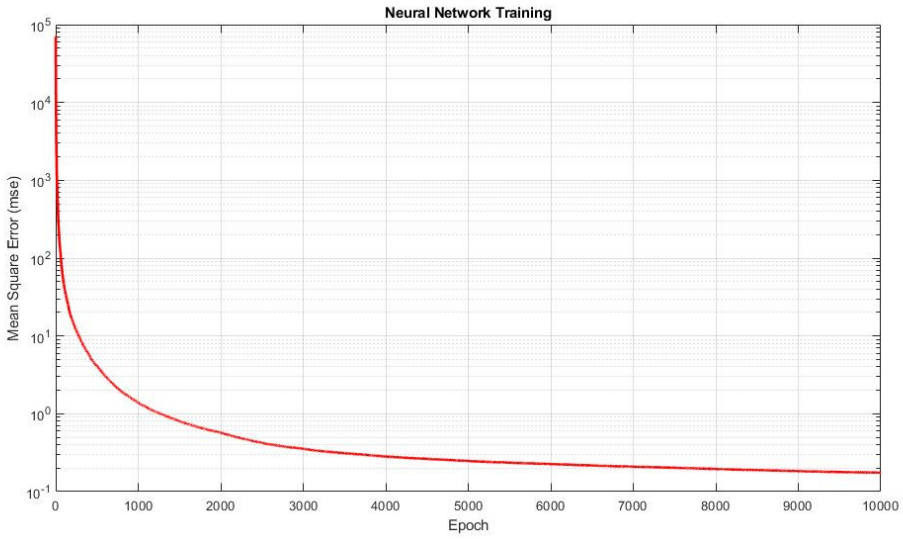

Fig. 5 Mean-square error for the artificial neural network.

utilization of this ANN for a formalized selection of weights for the sequential algorithm for cislunar transfers, as also for an adaptive strategy in the lines of what is described in the previous sub-section.

IV. Cascaded Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning focused on sequential decision-making in dynamic environments; hence, we have a natural motivation to compare the solutions generated by the RL framework with that obtained by the previously discussed sequential framework. We have previously developed a cascaded RL framework for computing orbit-raising trajectories to GEO [28], and are currently investigating the extension of the framework for studying orbit-raising maneuvers to NRHO. In general, a RL agent utilizes a Markov Decision Process (MDP) model that takes into account agent/environmental states, possible set of actions taken by the agent, and rewards associated with state-action pairs. The RL agent aims to maximize accumulated rewards by interacting with the environment, taking actions, receiving rewards, and transitioning between states. The agent's strategy is captured by a policy that maps states to action probabilities. In deep RL, a stochastic policy is often implemented through an actor-network. The actor-network's weights and biases determine the policy. In addition, a value function, used to assess state or state-action importance, estimates the expected sum of rewards when taking action at a current state under a given policy. The critic network, with weights and biases, is responsible for approximating this function. Actor-critic RL algorithms integrate policy (actor) and value (critic) networks. They collaboratively update at each training step, with the policy network maximizing expected rewards through policy gradient estimation, and the value network minimizing the mean squared Bellman error. The standard objective function seeks to maximize the expected sum of rewards under policy. The optimization process involves iterative updates to the policy and value networks until convergence. In this section, we describe a reinforcement learning framework for computing orbit-raising trajectories to NRHO.

A. RL Framework for Orbit-Raising Problem

Similar to the sequential methodology, the spacecraft dynamics is described by treating the angle ϕ as the independent variable (in contrast to the real independent variable time). While the current work focuses on minimum-time transfers, we have the aim of development of a framework that allows for computation of minimum-propellant transfers as well as incorporation of ephemeris-based models for capturing effects of gravitation exerted by third (or other) bodies. Hence, we use a state-vector $\boldsymbol{s}$ for RL implementation consisting of the (h, e) elements, described in the $\boldsymbol{x}$ vector (equation 1), as well as the spacecraft mass m, and time t. However, the current focus being minimum time transfers, the action is defined as the two angles that define the thrust direction, the in-plane angle α and the out-of-plane angle β. This imples that the spacecraft keeps the constant maximum thrust value throughout the transfer. Of course, in the presence of eclipses, the available thrust will be zero; however, eclipse effects are not considered as part of the present study. We also consider that the angles α and β are bounded in the ranges of $[-\pi, \pi]$ and $[-\pi/2, \pi/2]$ respectively. In other words, the set of possible actions that the spacecraft can take at each step is given by:

$$\boldsymbol{A} = \begin{bmatrix} \alpha & \beta \end{bmatrix}^T \tag{7}$$

Three critical terminal conditions are evaluated at each time step to determine if the spacecraft has reached its target; these conditions are based on the spacecraft's orbital elements, specifically the eccentricity (e), semi-major axis (a), and inclination (i):

1) Eccentricity Check:

$$e_{\text{tar}} - e_{\text{tol}} \leq \left[e = \sqrt{e_x^2 + e_y^2} \right] \leq e_{\text{tar}} + e_{\text{tol}}$$

2) Semi-Major Axis Check:

$$a_{\text{tar}} - \text{a}_{\text{tol}} \leq \left[a = \frac{h^2}{\mu(1 - e_x^2 - e_y^2)} \right] \leq a_{\text{tar}} + a_{\text{tol}}$$

3) Inclination Check:

$$i_{\text{tar}} - i_{\text{tol}} \leq \left[i = \cos^{-1}\left(\frac{\sqrt{h_x^2 + h_y^2}}{h} \right) \right] \leq i_{\text{tar}} + i_{\text{tol}}$$

Here, 'tar' represents target, and 'tol' represents tolerance.

B. Soft Actor-Critic Algorithm

The Soft Actor-Critic (SAC) algorithm, an advanced off-policy method, is recognized for its effectiveness in real-world RL applications. Unlike on-policy algorithms, SAC learns the target policy from a replay buffer containing

trajectories from a different behavioral policy. The key innovation of SAC is addressing the exploration-exploitation dilemma by introducing an entropy regularization term ($\mathbb{H}$) into the objective function. This encourages the agent to explore promising regions more thoroughly, while also avoiding over-fitting. The SAC objective function, incorporating entropy, is defined as:

$$J(\pi) = \mathbb{E}_{\boldsymbol{A}_t \sim \pi(.|\boldsymbol{s}_t)} \left[\sum_t \gamma^t (r(\boldsymbol{s}_t, \boldsymbol{A}_t) + \alpha \mathbb{H}(\pi(.|\boldsymbol{s}_t)) \right] \tag{8}$$

The control coefficient α modulates the level of entropy in the policy. SAC employs three networks: a policy (actor) network parameterized by θ, two Q-functions (critic) with parameters ϕ_1 and ϕ_2, and two target Q-functions with parameters ψ_1 and ψ_2. Using two Q-functions addresses the overestimation issue, ensuring reliable Q-value estimation. The loss function for Q-networks is given by:

$$\mathbb{L}_{\phi_i} = \frac{1}{|B|} \sum_{(s,A,r,s',d) \in B} \left(Q_{\phi_i}(s, A) - y(r, s', d) \right)^2 \tag{9}$$

where $y(r, s', d)$ is calculated through target Q-networks Q_ψ in eq (10).

$$y(r, s', d) = r + \gamma(1 - d) \left(\min_{i=1,2} Q_{\psi_i}(s', A'_\theta) - \alpha \log \pi(A'_\theta | s) \right), \quad A'_\theta \sim \pi_\theta(\cdot | s') \tag{10}$$

The policy loss function is described as:

$$\mathbb{L}_\theta = \frac{1}{|B|} \sum_{s \in B} \left(\underset{j=1,2}{min} Q_{\phi_j}(s, A_\theta(s)) - \alpha log \pi_\theta(A_\theta(s)|s) \right) \tag{11}$$

Target Q-networks (ψ_k) are updated at a slower rate than Q-functions (ϕ_k) to enhance training stability. This update is governed by the equation:

$$\psi_k \leftarrow p\psi_k + (1 - p)\phi_k \qquad for \;\; k = 1, 2$$

C. Cascaded Reinforcement Learning

We introduce a novel approach, which we refer to as cascaded deep reinforcement learning (DRL), where multiple independent reinforcement learning agents sequentially contribute to solving a complex problem. Each agent, implemented using Soft Actor-Critic (SAC) networks, addresses a sub-problem, breaking down the overall task into more manageable intermediate goals. In our study, we employ two SAC networks to achieve a target position with higher accuracy and resolution compared to a single-agent approach. The first network guides the agent to an intermediate goal defined by a relaxed set of terminating conditions, and enabling efficient exploration of the state space. The second network refines the policy for achieving the final target position with enhanced precision on the terminal conditions. This cascaded DRL approach offers advantages by promoting efficient exploration, enabling the learning of robust policies through the accomplishment of intermediate goals. While we recognize that designing effective task decompositions between the two DRL agents can be a good problem to investigate, currently we task assignment in an *ad hoc* manner, still providing advantages over a single agent DRL by reducing the overall training time, as well as ensuring a relatively more precise terminal condition.

A potential-based ($\phi(s') - \phi(s)$) shaping reward function is applied to train the DRL agents. This reward function includes a distance-based heuristic and a shaping function that contains the gradient information of the parameter's error values. The shaping function is a negative-powered exponential factor that becomes active when the difference between the current and target parameters is sufficiently small. The reward function is defined in Eq. (12) and Eq. (13) below:

$$\phi(os_t) = -w_1(|os_t - os_{tar}|) + \sum w_2 e^{-(w_3|os_t - os_{tar}|)} \tag{12}$$

$$r' = \phi(os') - \phi(os) - \tau - 5(\xi) + 100(d) \tag{13}$$

Here, w_1, w_2, and w_3 are the weights, and τ is a constant value that accounts for time penalty, which punishes the agent as the episode duration becomes longer. Its value is set to 0.005 in this study. os refers to the orbital elements semimajor axis a, eccentricity e, and inclination i. ξ is the boundary flag, and it indicates if the agent crosses the

boundary conditions, which are also defined in terms of the orbital elements (a,e,i) and restrict their values within a defined range to encourage the agent to explore in the right direction. The ξ is set to one when the agent crosses these boundary conditions, indicating an unsuccessful episode. The done flag d is set to zero and it becomes one only when the agent reaches the target destination, and zero otherwise. The negative reward function in Eq. (12) reinforces the desired behavior of the agent. The parameters chosen for calculating the reward values in Eq. (12), play a crucial role in the early convergence of the algorithm. In this study, the orbital elements, defined in terminal conditions, are used as the reward parameters. Eq. (13) finds the gradient of state parameters by essentially calculating the difference of state functions, which guides the agent towards the desired destination. Compared to the cascaded DRL development in our previous study on orbit-raising to GEO [28], we encountered significant challenges in achieving an optimized transfer trajectory in the presence of two celestial bodies, which leads to a more complex dynamic environment. In order to address the challenge, we adopted a backward propagation approach, similar to the optimization-based approaches.

Example 5. Transfer from a super-GTO to a NRHO.

Setting the initial position on the NRHO, our goal was to construct the trajectory leading to the super-GTO. This approach allowed for a more flexible and efficient trajectory planning process. Our implementation framework for optimizing the space trajectory in the orbit-raising problem involves dividing the task into two subtasks. The first subtask focuses on transporting the spacecraft in the vicinity of the target position (albeit through backward propagation), while the second subtask is dedicated to achieving the final target position with a relatively higher precision. To accomplish this, we implemented two sequentially connected SAC reinforcement learning algorithms. Table 2 summarizes the DRL terminal condition tolerances as well as the boundary conditions for the initial and final states for the DRL-based methodology. Figure 6 summarizes these results. Experiments were conducted on an Intel(R) Xeon(R) CPU E5-1620

Table 2 DRL parameters for a transfer from super-GTO to NRHO transfer.

Networks	Tolerance: a (km)	Terminating e	Elements i (deg)	Transfer Time (days)
First DRL	10	0.01	4.5	28.46
Second DRL	1	0.001	0.09	2.32

Boundary Conditions	h (km^2/s)	h_x (km^2/s)	h_y (km^2/s)	e_x	e_y	m (kg)
DRL (initial)	384073.943	43240.356	-111563.294	-0.0209	-0.1249	830
DRL (final)	101055.564	8993.124	-44988.209	0.6342	0.1422	1000

v4 operating at 3.5GHz with 8 cores, along with the NVIDIA GeForce GTX 1080 GPU and 32GB of RAM to meet computational requirements. Implementation was carried out in Python 3.7 using the PyTorch framework based on OpenAI stable baselines. This hardware/software combination facilitated efficient model development and training. DRL algorithms underwent multiple iterations/episodes to optimize performance, ensuring stable reward values for successful episodes in orbit-raising scenarios. Training continued until stability was achieved, typically requiring a maximum of 900 episodes. The selection criterion for optimal model weights was based on the minimum episodic convergence time, ensuring superior performance during the inference phase. These carefully chosen weights were utilized to generate optimized transfer time trajectories. To ensure the consistency and reproducibility of our findings, we conducted multiple trials during the training phase. Verification confirmed that reported results remained consistent across all episodes during the inference stage, reinforcing the validity of our research outcomes.

We present the findings of our approach, which involves training two cascaded DRL agents to optimize both the maximum episodic return (score) and minimum transfer time during the complete transfer training. Table 2 provides the convergence times and tolerance values for both DRL agents. The final transfer time of the trajectory is the cumulative result of the two DRL networks' efforts. The primary objective in determining convergence values for the two DRL networks is twofold. First, the initial network aims to guide the spacecraft to the vicinity of the target position, and second, the subsequent network refines the trajectory with higher resolution and precision to achieve the target value. This approach significantly contributes to achieving an improved tolerance range convergence. Figure 7 illustrates a decrease in training score variation as both cascaded DRL networks converge to a stable score value, with both networks

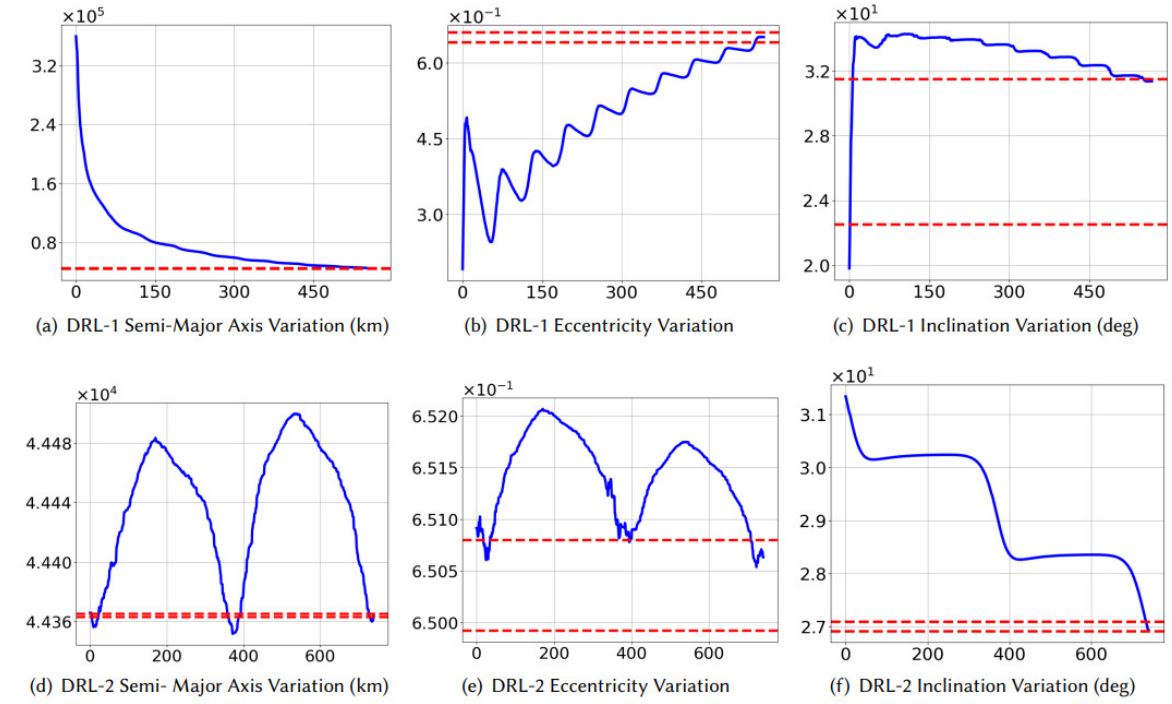

Fig. 6 Terminal condition satisfaction for DRL 1 and 2.

reaching convergence in under 800 episodes. Finally, the trajectory obtained by the cascaded DRL methodology yields a

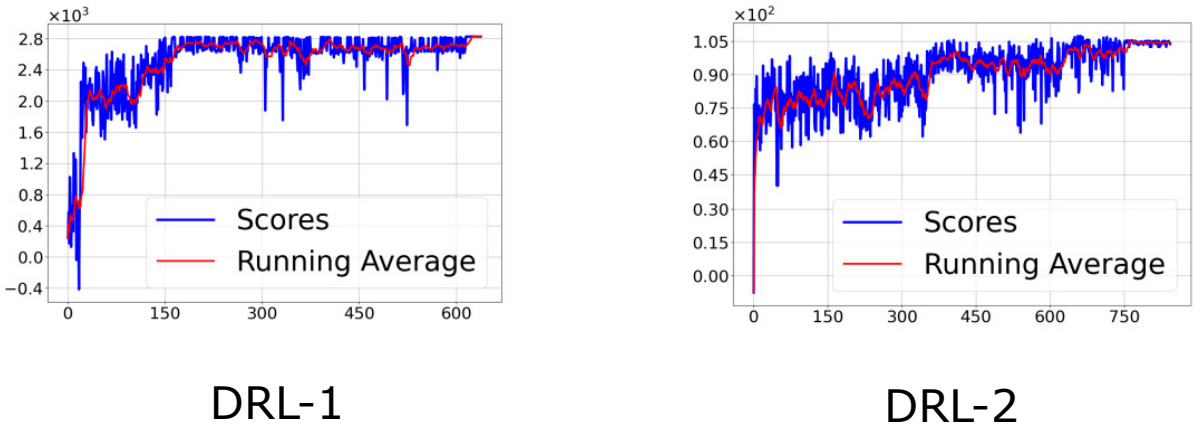

Fig. 7 Training score variation for DRL 1 and 2.

trajectory of 30.48 days, which is better than what is obtained by the sequential methodology, but which is sub-optimal (around 1.5 days longer) compared to the local minimum solution obtained by optimizing the trajectory yielded by the sequential algorithm. The trajectory is depicted in Figure 8. Please note that the tolerance on the terminal condition are a bit relaxed compared to the procedure for the sequential solution followed by optimization. Future work will focus on the improvement of the DRL algorithm in obtaining a better quality solution, both in terms of resulting transfer time and tolerance on the terminal state. The current DRL approach assumes constant thrust angles (action variables) over a given state transition step; this will be relaxed in future studies.

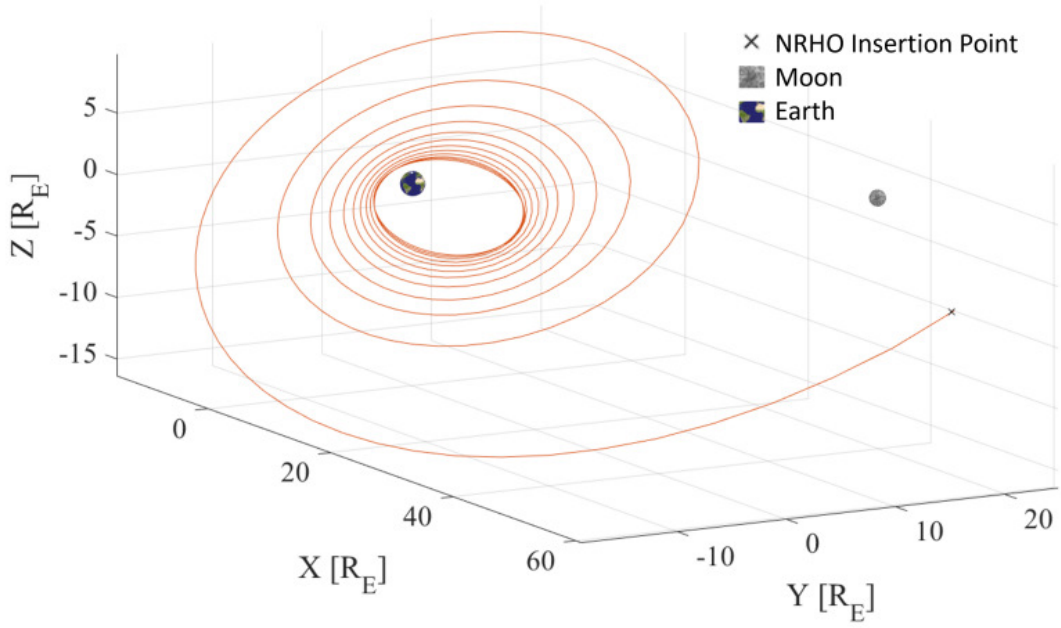

Fig. 8 Trajectory obtained by the cascaded DRL methodology.

V. Conclusion

The paper considers the problem of orbit-raising to different target orbits, such as the geosynchronous equatorial orbit and the near-rectilinear halo orbit. The sequential algorithm is a fast and robust way of automated computation of low-thrust orbit-raising trajectory computation. While the trajectory is computed in a convenient manner, the solution is sub-optimal. Augmented by artificial neural networks that can predict the transfer time for a transfer to GEO, an adaptive weight selection strategy can be used to improve upon the basic solution provided by the sequential algorithm. This improvement comes at the cost of additional offline training time needed before the execution of the sequential algorithm. While substantial improvement is achieved, the current implementation of the adaptive strategy considers dataset generated by the sequential algorithm. Obviously, it would benefit if the neural network is trained on globally optimal or near-optimal solutions. To this end, the paper presents some initial results on the training of artificial neural networks on dataset consisting of good quality locally optimal solutions by performing end-to-end optimization around the sequential algorithm. An alternative approach of using a cascaded reinforcement learning algorithm for computing the orbit-raising trajectory to near-rectilinear halo orbit is also presented. The reinforcement learning framework yields a solution better than the sequential approach, and this improvement comes at the cost of training two agents in a sequential manner. The solution of the deep reinforcement learning framework is sub-optimal; one source of the sub-optimality is the consideration of constant actions (thrust angles) during a transition step between two states within the reinforcement learning framework. In our previous work, we have shown that the cascaded deep reinforcement learning approach leads to near-optimal solution for orbit-raising to GEO; for the case of cislunar transfers, future work will therefore focus on improving the current framework by an alternate modeling that allows the thrust angles to vary during each state transition step.

References

[1] Lawden, D., Optimal Trajectories for Space Navigation, Butterworths, London, 1963.

[2] Kiforenko, B. N., "Optimal low-thrust orbital transfers in a central gravity field," International Applied Mechanics, Vol. 41, No. 11, 2005, pp. 1211–1238. https://doi.org/10.1007/s10778-006-0028-9, URL http://dx.doi.org/10.1007/s10778-006-0028-9.

[3] Russell, R., "Primer Vector Theory Applied to Global Low-Thrust Trade Space Studies," Journal of Guidance, Control and Dynamics, Vol. 30, No. 2, 2007, p. 2.

[4] Frankowska, H., and Vinter, R., "Existence of neighboring feasible trajectories: applications to dynamic programming for state-constrained optimal control problems," Journal of Optimization Theory and Applications, Vol. 104, No. 1, 2000, pp. 20–40.

[5] Falck, R., and Dankanich, J., "Optimization of Low-Thrust Spiral Trajectories by Collocation," AIAA/AAS Astrodynamics Specialist Conference, Guidance, Navigation, and Control and Co-located Conferences, Minneapolis, MN, 2012.

[6] Betts, J. T., Campbell, S. L., and Thompson, K. C., "Solving optimal control problems with control delays using direct transcription," Applied Numerical Mathematics, Vol. 108, 2016, pp. 185–203.

[7] Kluever, C. A., and Oleson, S. R., "Direct approach for computing near-optimal low-thrust earth-orbit transfers," Journal of Spacecraft and Rockets, Vol. 35, No. 4, 1998, pp. 509–515.

[8] Dutta, A., Libraro, P., Kasdin, N. J., and Choueiri, E., "A Direct Optimization Based Tool to Determine Orbit-Raising Trajectories to GEO for All-Electric Telecommunication Satellites," AAS Astrodynamics Specialist Conference, Minneapolis, MN, 2012.

[9] Betts, J. T., Campbell, S. L., and Digirolamo, C., "Initial guess sensitivity in computational optimal control problems," Numerical Algebra, Control & Optimization, Vol. 10, No. 1, 2020, p. 39.

[10] Wall, B., and Conway, B., "Shape-Based Approach to Low-Thrust Rendezvous Trajectory Design," Journal of Guidance, Control, and Dynamics, Vol. 32, No. 1, 2009, pp. 95–101.

[11] Taheri, E., and Abdelkhalik, O., "Shape-Based Approximation of Constrained Low-Thrust Space Trajectories Using Fourier Series," Journal of Spacecraft and Rockets, Vol. 49, No. 3, 2012, pp. 535–545.

[12] Trélat, E., "Optimal control and applications to aerospace: some results and challenges," Journal of Optimization Theory and Applications, Vol. 154, No. 3, 2012, pp. 713–758.

[13] Chilan, C. M., and Conway, B. A., "A reachable set analysis method for generating near-optimal trajectories of constrained multiphase systems," Journal of Optimization Theory and Applications, Vol. 167, No. 1, 2015, pp. 161–194.

[14] Kluever, C., "Simple Guidance Scheme for Low-Thrust Orbit Transfers," Journal of Guidance, Control, and Dynamics, Vol. 21, No. 6, 1998, pp. 1015–1017.

[15] Petropoulos, A. E., "Simple Control Laws for Low-thrust Orbit Transfers," AIAA/AAS Astrodynamics Specialist Conference, Big Sky, MT, 2003.

[16] Flandro, G. A., "Asymptotic Solution for Solar Electric Low-Thrust Orbit Raising with Eclipse Penalty," AIAA Mechanics and Control of Flight Conference, Anaheim, CA, 1974.

[17] Wiesel, W., and Alfano, S., "Optimal Many-Revolution Orbit Transfer," Journal of Guidance, Control and Dynamics, Vol. 8, No. 1, 1985, pp. 155–157.

[18] Kechichian, J. A., "Low-Thrust Eccentricity-Constrained Orbit Raising," Journal of Spacecraft and Rockets, Vol. 35, No. 3, 1998, pp. 327–335.

[19] Kechichian, J. A., "Orbit Raising with Low-Thrust Tangential Acceleration in Presence of Earth Shadow," Journal of Spacecraft and Rockets, Vol. 35, No. 4, 1998, pp. 516–525.

[20] Kechichian, J. A., "Low-Thrust Inclination Control in Presence of Earth Shadow," Journal of Spacecraft and Rockets, Vol. 35, No. 4, 1998, pp. 526–532.

[21] Colasurdo, G., and Casalino, L., "Optimal Low-Thrust Maneuvers in Presence of Earth Shadow," AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Providence, RI, 2004.

[22] Kluever, C., "Using Edelbaums Method to Compute Low-Thrust Transfers with Earth-Shadow Eclipses," Journal of Guidance, Control and Dynamics, Vol. 34, No. 1, 2011, pp. 300–303.

[23] Falck, R. D., Sjauw, W. K., and Smith, D. A., "Comparison of Low-Thrust Control Laws for Applications in Planetocentric Space," 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Cleveland, OH, 2014.

[24] Junkins, J. L., and Taheri, E., "Exploration of Alternative State Vector Choices for Low-Thrust Trajectory Optimization," Journal of Guidance, Control, and Dynamics, Vol. 42, No. 1, 2019, pp. 47–64.

[25] Miller, D., and Linares, R., "Low-Thrust Optimal Control Via Reinforcement Learning," AAS/AIAA Astrodynamics Specialist Conference, 2019.

[26] LaFarge, N. B., Miller, D., Howell, K. C., and Linares, R., "Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment," Acta Astronautica, Vol. 186, 2021, pp. 1–23. https://doi.org/https://doi.org/10.1016/j.actaastro.2021.05.014, URL https://www.sciencedirect.com/science/article/pii/S0094576521002460.

[27] Yanagida, K., Ozaki, N., and Funase, R., Exploration of Long Time-of-Flight Three-Body Transfers Using Deep Reinforcement Learning, ???? https://doi.org/10.2514/6.2020-0460, URL https://arc.aiaa.org/doi/abs/10.2514/6.2020-0460.

[28] Zaidi, S. M. T., Chadalavada, P. S., Ullah, H., Munir, A., and Dutta, A., "Cascaded Deep Reinforcement Learning-Based Multi-Revolution Low-Thrust Spacecraft Orbit-Transfer," IEEE Access, Vol. 11, 2023, pp. 82894–82911. https://doi.org/10.1109/ACCESS.2023.3301726.

[29] Bosanac, N., Bonasera, S., Sullivan, C., McMahon, J., and Ahmed, N., "Reinforcement Learning for Reconfiguration Maneuver Design in Multi-Body Systems," AAS/AIAA Astrodynamics Specialist Conference, Virtual, 2021.

[30] Sullivan, C., Bosanac, N., Mashiku, A., and Anderson, R., "Multi-Objective Reinforcement Learning for Low-Thrust Transfer Design between Libration Point Orbits," AAS/AIAA Astrodynamics Specialist Conference, Virtual, 2021.

[31] Holt, H., Baresi, N., and Armellin, R., "Towards Optimal Lyapunov Controllers for Low-Thrust Transfers Via Reinforcement Learning," AAS/AIAA Astrodynamics Specialist Conference, Virtual, 2021.

[32] Holt, H., Armellin, R., Baresi, N., Hashida, Y., Turconi, A., Scorsoglio, A., and Furfaro, R., "Optimal Q-laws via reinforcement learning with guaranteed stability," Acta Astronautica, Vol. 187, 2021, pp. 511–528. https://doi.org/https://doi.org/10.1016/j.actaastro.2021.07.010, URL https://www.sciencedirect.com/science/article/pii/S0094576521003684.

[33] Sreesawet, S., and Dutta, A., "Fast and Robust Computation of Low-Thrust Orbit-Raising Trajectories," Journal of Guidance, Control, and Dynamics, Vol. 41, No. 9, 2018, pp. 1888–1905.

[34] Arora, L., and Dutta, A., "Reinforcement Learning for Sequential Low-Thrust Orbit Raising Problem," AIAA Scitech 2020 Forum, 2020, p. 2186.

[35] Chadalavada, P., and Dutta, A., "Selecting Planning Horizon Length for Sequential Low-Thrust Orbit-Raising Optimization Problem," AAS/AIAA Astrodynamics Specialist Conference, Portland, ME, 2019.

[36] Chadalavada, P., Farabi, T., and Dutta, A., "Sequential Low-Thrust Orbit-Raising of All-Electric Satellites," Aerospace, Vol. 7, No. 6, 2020, p. 74.

[37] Arustei, A., and Dutta, A., "An adjoint sensitivity method for the sequential low-thrust orbit raising problem," AAS/AIAA Astrodynamics Specialist Conference, Charlotte, NC, 2022.